基于NoC结构的图像中值滤波并行处理模式分析

刘 佳[1] 路 铭[2] 李哲英[1]
(北京联合大学信息学院 北京 100101)[1] (北京联合大学应用科技学院 北京 100101)[2]

摘 要 给出了一个多处理器NoC结构以实现指定的中值滤波算法。为了提高图像处理的速度,在NoC设计的专用SoC中使用了系统并行机制与基本计算单元指令并行机制相结合的方法。它既可以满足处理速度的要求,又能达到降低功率损耗的目的。对图像处理中的中值滤波处理结构进行了并行设计,可极大地提高处理速度。

关键词 NoC,中值滤波,并行处理
中图法分类号 TP242.6 **文献标识码** A

Model Analysis of Image Median Filter Based on NoC

LIU Jia[1] LU Ming[2] LI Zhe-ying[1]
(College of Information, Beijing Union University, Beijing 100101, China)[1]
(College of Applied Science Technology, Beijing Union University, Beijing 100101, China)[2]

Abstract The paper proposed a NoC architecture for realizing a special median filter algorithm. In order to improve the processing speed, we combined the system-level parallel processing mechanism and instruction-level parallel processing mechanism in a special SoC, which not only can satisfy the processing speed, but also can lower the power consumption.

Keywords NoC, Median filter, Parallel processing

1 引言

近年来,随着信息处理技术和微电子技术的发展,具有多处理器系统的MPSoC,正逐步成为图像处理系统的核心技术。MPSoC实际上就是一个可以并行执行工作的片上网络NoC,利用NoC技术可以实现高速并行的数据处理系统。

为了提高图像处理系统的质量,降低对机器视觉数据处理系统的速度的过高要求,近年来开始采用从数据源进行某些预处理的技术。文献[1,2]提出在CCD中采用嵌入式的模拟存储和相应的时空滤波来提高图像质量,以降低光线等因素造成的图像模糊。文献[3]提出了低电平图像处理的方法,在CCD中嵌入了low-level图像处理单元,这种处理采用了海量并行方式,从而提高了处理速度和图像质量,并降低了系统的图像处理要求和功率损耗。不过,在文献[1-3]等提供的技术中,其像素点数一般都比较少。随着机器视觉中图像处理速度的提高,系统功率损耗也增加了。如何在现有机器视觉图像处理系统中提高图像预处理的速度,同时降低系统功率损耗,是目前机器视觉技术所面临的挑战[4-6]。

基于现有CCD或CMOS系统的非嵌入式并行中值处理结构,可以有效提高处理速度并降低系统的功率损耗[6]。NoC由于具有巨大的处理资源,可以实现对数据处理的并行算法,使用并行方式进行图像处理,既可以满足处理速度的要求,又能达到降低功率损耗的目的。对图像处理中的中值滤波处理结构进行了并行设计,可以极大地提高处理速度。

本文第2节讨论了中值滤波的并行处理;第3节讨论了中值滤波并行算法的实现结构;第4节给出了实现中值滤波并行算法的具体计算,并对其结果进行了简单的讨论。

2 图像中值滤波处理

图像中值滤波实际上是一种空间滤波方法,目的是提高图像质量和清晰度,为后续的图像处理提供质量良好的图像数据。

2.1 中值滤波基本算法

在图像处理中,中值滤波的目的是对图像进行平滑滤波,以去除图像中的尖峰噪声,从而为进一步的处理提供比较清晰的图像。

设 f_i 是数字信号序列,中值滤波器的处理定义如下:

$$Y_i=Med\{f_{i-v}\cdots f_i\cdots f_{i+v}\}$$
$$v=\frac{m-1}{2} \quad (1)$$

式中,m 是中值处理的数据个数,实际计算中 m 是奇数,v 为中值半径,代表最远取数距离。

从式(1)可知,中值 Y_i 就是在 m 个数据中找出一个中间值(不是平均值)以代替 f_i。对于2维图像数据,中值滤波需要按一定的形状处理,例如图1所示的菱形(rhombus),也可以按其他形状处理。

中值滤波处理的目的,是要找到 a 个数据按从小到大顺

到稿日期:2011-02-23 返修日期:2011-05-24 本文受北京市教育委员会科技面上项目图像处理专用NoC算法与结构研究(PHR(IHLB)20090513),北京市优秀人才培养资助项目多处理器NoC结构研究(2010D005022000011)资助。

刘 佳(1978-),女,硕士生,讲师,主要研究方向为图像处理算法研究,E-mail:xxtliujia@buu.edu.cn;路 铭(1981),男,硕士生,助教,主要研究方向为集成电路;李哲英(1953-),男,硕士,教授,主要研究方向为集成电路、算法研究。

序排列时处于数据排列中间位置的数据，因此，图 1 所示的 5 个数据的中值检测中，只需要进行到 x_i 即可。就是说，把数据排列成

$$A \leqslant B \leqslant C \begin{cases} \leqslant D \\ \leqslant E \end{cases}$$

由此可知，只要 $C<D$，并且小于 E，则只要找到 C 即可。

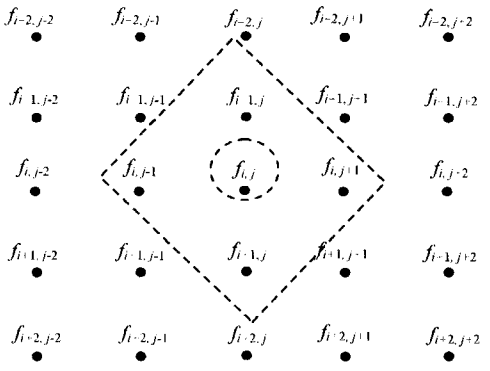

图 1　图像像素分布与中值滤波示意

设 a 为每次处理的数据个数，从 $i-v$ 到 $i+v$ 的计算如下：

$$\begin{cases} x_1 = \min\{f_{i-v}, \cdots, f_{i+v}\} \\ x_2 = \min\{f_{i-v+1}, \cdots, f_{i+v}\} \\ \quad \vdots \\ x_i = \min\{f_i, \cdots, f_{i+v}\} \end{cases} \tag{2}$$

根据式(2)，处理过程中要对数据进行循环方式的两两比较，每次选择出两个数据中最小的一个继续与其它数据进行比较；同时，还要保留比较结果中最大的数，以便能继续与其他数据进行比较。因此，根据式(2)，每一个像素点的中值滤波需要 9 次比较计算处理。根据中值滤波的定义，“查找 n 个数中处于中间位置上的数”，由于 $x_i \leqslant \{f_{i+1}, \cdots, f_{i+v}\}$，因此之后的比较已经没有意义了。

进行中值查找时，首先需要确定处理方式，一种是覆盖方式，一种是非覆盖方式。

1)覆盖方式时，所处理的数据结果直接写入原始矩阵，因此，滤波过程具有传递效应。

2)非覆盖方式时，所处理的数据形成新的矩阵，并不覆盖原始矩阵，因此，滤波效果仅与原始图像有关。

由于图像数据量很大，因此，尽量减少计算次数对提高系统工作速度、降低系统功率损耗具有十分重要的意义。

2.2　并行处理算法时间分析

道路识别图像处理系统中，由于边沿提取和道路识别需要使用比较复杂的算法，因此，要求中值滤波能够尽量提高工作速度，同时，还要求中值滤波部分尽可能地降低系统功率损耗。为此，需要对中值滤波的算法进行分析，采用尽可能高的并行处理算法。

设中值滤波中，图像矩阵为 PQ，每个点处理计算需要 N 条指令，读取一个原始数据需要 n 条指令，指令时钟频率为 f_s，则单一处理器对一幅图像进行中值滤波的时间为：

$$T_m = \frac{PQ(N+an)}{f_s} \tag{3}$$

设所使用的微处理器具有指令并行机制，中值滤波计算中存在指令并行系数 η，这个系数代表了实际操作所需要的指令数与处理所需全部指令的比值，即：

$$\eta = \frac{(N+an)-N_p}{N+an} \leqslant 1 \tag{4}$$

式中，Np 是算法中可并行执行的指令数。因此，

$$T_m = \frac{PQ\eta(N+an)}{f_s} \tag{5}$$

如果使用 M 个基本处理单元完成中值滤波，考虑到每个基本处理单元接收完 a 个数据后立即开始处理，而与此同时，下一个基本处理单元开始接收数据，所以 M 个基本处理单元完成 M 个像素点的中值滤波处理所需要的时间为：

$$T_{pM} = \frac{anM + \eta_M N}{f_{pM}} \tag{6}$$

注意，在最后一个基本处理单元接收完数据后，还需要对数据进行处理，同时，在最后一个处理单元完成像素处理之前，其他所有的处理单元都已经完成了处理工作。

2.3　确定 M 的方法

在设计并行中值滤波处理的 NoC 时，选择基本处理单元个数 M 时要注意以下 3 个因素。

1)整数条件。设中值滤波处理采用对图像逐行扫描，对具有 P 行、Q 列的图像来说，基本处理单元的个数应当满足如下条件：

$$\frac{Q}{M} = \text{整数} \tag{7}$$

之所以这样要求，是为了保证每一个基本处理单元都能处理相同数量的像素点，并能够使系统工作更具逻辑性。

2)指令比例。在选择 M 时，除了式(7)的限制条件外，还必须考虑像素处理指令时间与 M 个基本处理单元读取原始数据的比例，即：

$$\xi = \frac{anM}{\eta_M N} \geqslant 1, n \in M \tag{8}$$

这就是要求，最后一个基本处理单元读取完数据后，第一个基本处理单元已经完成像素处理，可以接收新数据。ξ 应当尽量接近 1，这样可以减少第一个基本处理单元在处理完一个像素后的等待时间，如果 $\xi=1$，则 M 个基本处理单元都可以连续工作而没有等待时间。

3)时间限制。对于 M 的另一个限制要求，则是在给定指令周期的条件下，满足图像中值滤波处理所需的时间要求。设 T_{im} 是系统要求的完成一幅图像中值滤波处理的时间，T_M 是 M 个基本处理单元完成一幅图像处理所需要的时间，则 M 必须满足：

$$T_{im} \geqslant T_M \tag{9}$$

3　中值滤波并行算法结构

为了对并行机制分析，这里建立了中值滤波处理的数据流图模型，如图 2 所示。在图 2 中，激发节点代表系统保存原始图像数据，A 是图像原始数据缓冲 RAM，C 代表基本处理单元，D 代表已处理数据缓冲 RAM。

从这个 DFG 可以看出，原始数据在一个基本处理单元的控制下，按一定的顺序写入缓冲 RAM，M 个中值滤波处理的基本处理单元完成中值滤波，并把处理结果保存在输出缓冲 RAM 中。此外，这种处理过程中没有使用覆盖方式，就是说，所有像素点的中值滤波都是以原始数据为基础进行的。

图 2 还指出，要实现完全并行，必须建立 A 到 C 之间的多条数据通道，以便能够把数据同时传送给基本处理单元。

从硬件结构上看，这几乎是不可能的，因为只有在 RAM 存在 M 个端口时，才能把数据同时送到 M 个基本处理单元中。由于图像数据保存在 RAM 中，而 RAM 不可能具有多个数据接口，因此必须考虑一种适当的数据读取方法，以便把待处理的数据分配给各个基本处理单元。

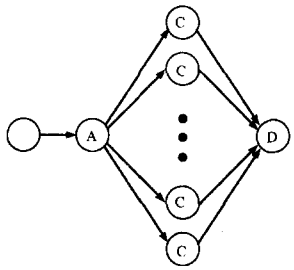

图 2　中值滤波处理 DFG

为了解决数据读取冲突问题，可以对图 2 进行更改，如图 3 所示。在图 3 中，增加一个基本处理单元 R 来控制原始图像数据的输入，同时设置了两个数据缓冲 RAM，即 A 和 B。这两个缓冲 RAM 的尺寸相同，每个都可以保存图像的 3 行数据。这两个缓冲 RAM 的任务是交替更新，基本处理单元在读取其中一个模块时，R 对另一个 RAM 模块进行更新。这样可以避免同一行数据处理时的 RAM 读取冲突。

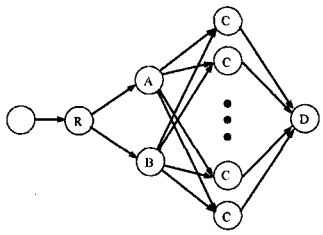

图 3　中值滤波处理 DFG

3.1　基本处理单元取数方式

本文中值滤波使用 $a=5$ 的十字数据选择方法，数据的读入规则如图 4 所示。

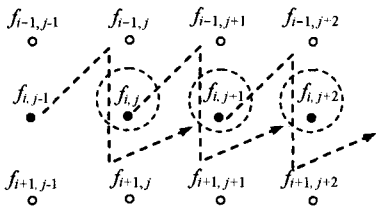

图 4　图像像素取数路线

由此可知，每个缓冲 RAM 要能够保存 3 行图像数据并且逐行更新。

在上述处理中，第 1 行和第 P 行处理时使用两次第 1 行和 $P-1$ 行的数据，第 1 列和第 Q 列处理时，使用两次第 1 列和 $Q-1$ 列的数据。图 5 给出了左上角像素和左下角像素处理数据的方法，其它边沿像素点处理数据的选取与此类似。图 5 中的实心点为实际像素点。

这里仅考虑灰度图像。设图像存储器中图像数据按行顺序排列，图像的列数为 Q，共有 P 行，所以，图像数据的存储 RAM 尺寸为 $QP\times$像素字节数。

对于图 4，考虑图像数据按行顺序存放（各行之间首尾相接），设被处理像素的地址是 $[f_{i,j}]$，则基本处理单元取数的顺序可以是 $[f_{i,j}-1]$，$[f_{i,j}]$，$[f_{i,j}+1]$，$[f_{i,j}+Q]$，$[f_{i,j}-Q]$。

如果令地址寄存器的首地址为 ADDR$=[f_{i,j}-1]$，在数据读取的过程中，ADDR 的内容不断变化，为了尽量减少对 ADDR 的处理指令，则取数时地址寄存器的变化顺序为

ADDR$=[f_{i,j}-1]$

ADDR$+1$

ADDR$+2$

ADDR$+Q-1$

ADDR$-2Q$

注意，上述中每一个 ADDR 中的地址数据都是上一个 ADDR 中的数据，例如，ADDR$+2$ 中，ADDR$=[f_{i,j}-1]+1$。

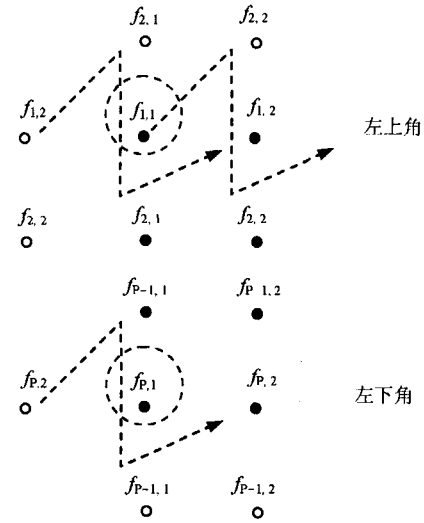

图 5　边沿像素处理方法

对于非边沿的数据，情况稍有不同。具体如表 1 所列。

表 1

左上角	左下角
ADDR$=[f_{1,1}]$	ADDR$=[f_{p,1}]$
ADDR$+1$	ADDR$+1$
ADDR$+1$	ADDR$+1$
ADDR$+Q-1$	ADDR$+Q-1$
ADDR$+Q-1$	ADDR$+Q-1$

同理，对于图像矩阵右侧的数据如表 2 所列。

表 2

右上角	右下角
ADDR$=[f_{1,Q-1}]$	ADDR$=[f_{p,Q-1}]$
ADDR$=[f_{1,Q-1}]$	ADDR$=[f_{p,Q-1}]$
ADDR$+1$	ADDR$+1$
ADDR$+Q$	ADDR$-Q$
ADDR$+Q$	ADDR$-Q$

4　并行处理算法的 NoC 结构特征

为了实现图 3 所示的快速并行算法，本文考虑处理系统的 NoC 结构具有如下特征。

1）系统提供两个具有相应存储容量的数据 RAM 模块，分别保存原始图像数据和处理结果数据。

2）基本处理单元具有 4 级流水线指令处理结构，$a=5$ 个数据（见图 4）时 $N=16$，取数据指令数 $n=1$，并行指令系数为 $\eta_M=0.75$，指令时钟为 200MHz。

3）基本处理单元具有相应数量的双向端口数据存储器。

此外，要求完成一幅 480 行、640 列图像处理的时间小于 10ms。

由以上特征可知，用来进行数据处理的基本处理单元具有较高的指令并行处理能力，为实行并行算法提供了支持。

5 算法实现平台

5.1 系统实现流程

在本文中，要完成 20frame/s 的实时动态识别，对于一帧 640 * 480 分辨率的视频图像，对其进行图像滤波运算处理必须尽快完成。

在硬件平台上实现实时运动图像中值滤波算法，需要完成以下内容，如图 6 所示。

1)使用 CCD 摄像头，在所设计的硬件平台上完成视频的实时采集；

2)连接标准 1280 * 1024 电脑显示器显示输出；

3)完成截取视频流，保存并读取一帧图像；

4)在读取下一帧图像之前，完成实时识别算法运算；

5)显示运动物体识别结果。

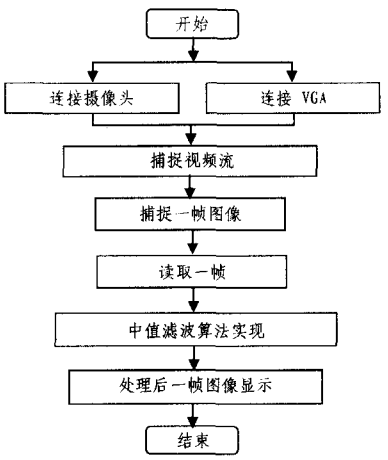

图 6 系统处理流图

5.2 NoC 系统结果分析

根据上述讨论，本文设计的 NoC 结构如图 7 所示。考虑取数据需要 10 条指令，而中值滤波处理需要 $\gamma_M N=0.75\times16=12$，根据式(8)，有：

$$\xi=\frac{5M}{12}\geqslant1$$

根据这个条件，$M=2.4$。如果选 $M=3$，则并行效果不明显，这里选择 $M=8$，所以 $k=80$。

图 7 中，E 代表基本处理单元，T 是 NoC 网络路由控制器，$M0$ 负责从 RAM 中获取数据并分配给相应的处理单元，$M1$ 则负责从个处理单元收集处理结构并存入 RAMb。虚线框中的基本处理单元用于中值滤波处理，其他的基本处理单元用于完成系统所需要的其他处理功能。

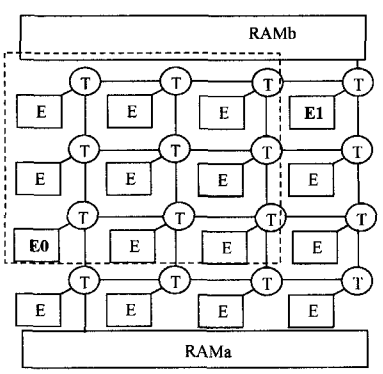

图 7 一个 4×4 的一般规则 NoC 结构

1)单一微处理器

作为对比，考虑在相同条件下，根据式(5)如果使用单一微处理器完成一幅图像的中值滤波处理，需要的时间是：

$$T_m=\frac{PQ\eta(N+an)}{f_s}=\frac{480\times640\times0.75(16+5)}{200\times10^6}$$

$$=24.192\text{ms}$$

2)NoC 结构处理器

设一幅图像有 PQ 个像素需要处理，参考式(6)，并行处理所需要的处理时间为：

一幅图像处理所需时间 T_M＝行数×每行处理所需时间－行间读数据的重叠时间。

根据第 2 节和第 3 节的分析，每一行的处理时间为：

$$T_{QM}=\frac{5Mk-5(k-1)+12}{200\times10^6}$$

式中，k 代表每个基本处理单元需要处理像素点的个数，$k\times M=Q$。

各行之间读数据的重叠时间为 $\frac{5(480-1)}{200\times10^6}$。把给定的各项数据代入上式，并考虑式(9)得：

$$10^{-5}\geqslant T_M=480\,\frac{5Mk-5(k-1)}{200\times10^6}-\frac{5(480-1)}{200\times10^6}+\frac{12}{200\times10^6}$$

$$\approx6.72\text{ms}$$

由此可知，当 $M=8$ 时，在给定的系统指令周期条件下，一幅图像的中值滤波处理时间仅为 6.72ms。

较单一微处理器，采用 NoC 结构，处理速度提高了 3.6 倍。

结束语 为了提高图像处理的速度，在 NoC 设计的专用 SoC 中使用了系统并行机制与基本计算单元指令并行机制相结合的方法。在并行中值滤波处理系统中，需要考虑指令时钟周期和像素点处理指令总数(包括并行指令)对基本处理单元数量的影响。所以，在设计的 NoC 结构中，并行与指令并行两种机制需要相互配合，才能得到最佳效果，达到提高处理速度的目的。

参 考 文 献

[1] Massari N, Gottardi M, Gonzo L, et al. A CMOS image sensor with programmable pixel-level analog processing[J]. IEEE Trans. Neural Networks, 2005, 16(6): 1673-1684

[2] Massari N, Gottardi M. A 100 dB Dynamic-Range CMOS Vision Sensor With Programmable Image Processing and Global Feature Extraction[J]. IEEE J. Solid-State Circuits, 2007, 42: 647-657

[3] Dubois J, Ginhac D, Paindavoine M, et al. 10000 fps CMOS Sensor With Massively Parallel Image Processing[J]. IEEE J. Solid-State Circuits, 2008, 43(3): 706-717

[4] Kozlowski L, Rossi G, Blanquart L, et al. Pixel noise suppression via SoC management of target reset in a 1920 * 1080 CMOS image sensor[J]. IEEE J. Solid-State Circuits, 2005, 40(12): 2766-2776

[5] Damera-Venkata N, Evans B L. Parallel implementation of multifilters[J]. Proc. IEEE ICASSP '2000, 2000, 6(6): 3335-3338

[6] Ehsan S, Clark A F, McDonald-Maier K D. Novel Hardware Algorithms for Row-Parallel Integral Image Calculation[C]// IEEE DICTA '09. 2009: 61-65